SIMONE GUGGIARI
ETH ZÜRICH 2016
LEGI # 14-932-867

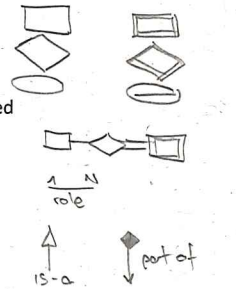
DATA MODELING AND DATABASES

1 INTRODUCTION

- what is a DB — data intensive shared
  - relational/graph
  - advantages — no redundancy/inconsistency
- data modeling
  - conceptual model — ER
  - logical model — schema
  - physical model

2 ENTITY RELATIONSHIP (ER) MODEL

- guidelines — aggregation (part-of) generalization (is-a)
- c4 (correct concise complete comprehensive)
- notation
  - entity (/weak)
  - relationship (/identifying)
  - attribute (/key)
    - multivalued/composite/derived
  - total participation
  - cardinality / min max
  - role
- UML



3 RELATIONAL MODEL (RM)

- Terms
  - relation R — $R \subseteq D_1 \times \dots \times D_n$
  - domain D
  - tuples — $t \in R$, unordered
  - schema — (associate labels to domains), table name, attribute (cols)
  - instance
  - (primary) key — (min set of attr that identify uniquely)
- Transformation ER->RM
  - 1,2) E, R → rel
  - 3) merge same key
  - 4) generalization
  - 5) weak entity
- Relational algebra
  - selection — $\sigma_p(E_1)$
  - projection — $\pi_S(E_1)$
  - Cartesian product — $E_1 \times E_2$
  - (inner) natural join — $R \bowtie S$ — $(R-S, R \cap S, S-R)$
  - theta join — $\bowtie_\theta$ — $\theta \in \{>, <, =, \dots\}$
  - rename — $\rho_V(E_1)$ relation, $\rho_{A \leftarrow B}(E_1)$ attribute
  - relation division
  - minus/union/intersection — $-$ $\cup$ $\cap$
  - semi join (left/right) — $\ltimes$ $\rtimes$

$t \in R \div S \Leftrightarrow \forall ts \in S\ \exists tr \in R: tr.S = ts.S \wedge tr.(R-S) = t$ → simulation

- outer join (full, left, right) — $\bowtie$ $\overrightarrow{\bowtie}$ $\overleftarrow{\bowtie}$
- Relational calculus — $\{t \mid P(t)\}$
  - tuple relation calculus
    - atoms
    - formulas
  - domain relation calculus — $\{[v_1, \dots, v_n] \mid P(v_1, \dots, v_n)\}$
- Codd's theorem → equivalence languages
  - rel algebra
  - tuple rel calc
  - dom rel calc

4 SQL EXAMPLES → see chapter

- Languages
  - DDL (data definition language)
  - DML (data manipulation language)
  - SQL (query language)
- Relational division
- NULL (state)
- recursion
- snapshot semantics → 1. work affected by update 2. update
- Views (store query)
  - logical data independence — privacy, access control

5 INTEGRITY CONSTRAINTS

- Transactions — Constraints: (pre/post conditions)
  - data correctness
- Referential Integrity Constraints
  - foreign key — (state/reference) — NULL valid ptr
  - OnUpdate / OnDelete
    - Cascade (propagate)
    - Restrict (no delete if reference)
    - No action (no violation)
    - Set default / Null
- Triggers — ECA Rule — event, condition → action

es. keys, multiplicity, attribute domain, subset, referential integrity

DB + APP: optimization, safety net, meaningful errors

6 FUNCTIONAL DEPENDENCIES (quality)

- Redundancy
- Algorithms
  - synthesis
  - decomposition
  - → see later
- Redundancy — bad
- Anomalies → extra work on redundancy → if not done, inconsistency (INSERT, UPDATE, DELETE)
- Functional dependencies
  - definition
  - determinant set (X)

TNDB: multi version DB → never throw away

schema $\mathcal{R} = \{A: D_A, B: D_B, C: D_C\}$, instance R, $\alpha, \beta \subseteq \mathcal{R}$

$\alpha \rightarrow \beta$ if $\forall r, s \in R: r.\alpha = s.\alpha \Rightarrow r.\beta = s.\beta$

(function $f: XD_\alpha \rightarrow YD_\beta$)

- dependent set (Y) — trivial $X \to Y$ (Y subset of X)
- Keys
  - superkey $\alpha \subseteq \mathcal{R} \Leftrightarrow \alpha \to \mathcal{R}$
  - minimal $\alpha \to \beta \Leftrightarrow \forall A \in \alpha: \neg((\alpha - \{A\}) \to \beta)$ — can't remove shit — notation $\alpha \to^{\circ} \beta$
  - candidate $\alpha \Leftrightarrow \alpha \to^{\circ} \mathcal{R}$
  - Cardinality
  - Properties
    - refl / augm / trans — $\beta \subseteq \alpha \Rightarrow \alpha \to \beta$ — $\alpha \to \beta \Rightarrow \alpha\gamma \to \beta\gamma$ — $\alpha \to \beta \wedge \beta \to \gamma \Rightarrow \alpha \to \gamma$
    - union / decomp / pseudo-trans — $\alpha \to \beta \wedge \alpha \to \gamma \Rightarrow \alpha \to \beta\gamma$ — $\alpha \to \beta\gamma \Rightarrow \alpha \to \beta \wedge \alpha \to \gamma$
  - Closure $\alpha \to \alpha^+$ — $\alpha \to \beta \wedge \gamma\beta \to \delta \Rightarrow \alpha\gamma \to \delta$
- Minimal basis F_c
  - condition — 1) $F_c \equiv F$ 2) minimal
  - computation (4) — 3) no same left side
    - reduction — $\forall A \in \alpha: (F_c - (\alpha \to \beta)) \cup ((\alpha - \{A\}) \to \beta)) \not\equiv F$
    - decomposition — $\forall B \in \beta: (F_c - (\alpha \to \beta)) \cup (\alpha \to (\beta - \{B\})) \not\equiv F$
  - lossless — $\mathcal{R} = \mathcal{R}_1 \bowtie \dots \bowtie \mathcal{R}_n$ (1,2) left & right side
  - preservation → $FD(\mathcal{R})^+ = (FD(R_1) \cup \dots \cup FD(R_n))^+$

$\alpha \to \beta$, $\beta \subseteq$ Closure $(F, \alpha - \{A\})$ then replace $(\alpha - \{A\}) \to \beta$

$\beta \subseteq (F - (\alpha \to \beta)) \cup (\alpha \to (\beta - \{B\}), \alpha)$ then replace $\alpha \to (\beta - \{B\})$

③ remove $\alpha \to \emptyset$

④ union same left side

$R = R_1 \cup R_2$, $\begin{pmatrix} R_1 = \Pi_{R_1}(R) \\ R_2 = \Pi_{R_2}(R) \end{pmatrix} \Leftrightarrow (R_1 \cap R_2) \to R_1$ or $\to R_2$

7 NORMAL FORMS

- idea
- 1NF
  - atomic domain (only)
- 2NF
  - minimal dependency
    - decomposition

= no non prime attribute is dependent on a proper subset of any candidate key

2NF "every non-key attribute minimally dependent on every key"

$\exists \alpha \to B: B$ not part of any key, $\alpha \subseteq K$, K is a key

a FD on any part of a candidate key violates 2NF

ER: N:M, 1:N

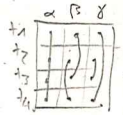
- 3NF
  - conditions
  - synthesis algorithm
    - steps 1-4
- BCNF
  - conditions
  - decomposition algorithm
- Multi-value dependencies (MVD)
  - definition
  - lossless
  - Laws
- 4NF
  - modified decomposition algorithm

$\Leftrightarrow \forall \alpha \to B$, one min holds: $B \in \alpha$ (trivial), B attribute of at least one key, α superkey of R

all $(B - \alpha)$ prime attribute of candidate key ← not in BCNF

in: R, FD F
1) compute F_c
2) $\forall \alpha \to \beta \in F$, create $R_\alpha := \alpha \cup \beta$
3) if $\exists k \subseteq R$ | k is key, create $R_k := k$
4) eliminate R_α if $\exists R_{\alpha'}$ with $R_\alpha \subseteq R_{\alpha'}$

$\alpha \to\!\!\to \beta$ (same, but with $\to\!\!\to$)



course	book	prof

*

result = {R}
while ($\exists R_i \in$ result : R_i not BCNF)
  let $\alpha \to \beta$ be evil in R_i
  $R_{i1} = \alpha \cup \beta$
  $R_{i2} = R_i - \beta$
  result = (result − R_i) ∪ R_{i1} ∪ R_{i2}
output (result)

not 4NF

$\to\!\!\to \beta$ be evil

8 SCHEMAS

- OLTP: TPC-C (online transaction processing) — update intensive — e-banking, shop
- OLAP: TPC-H (online analytical processing) — analysis
  - de-normalized schemas — read intensive
    - Star schema — fact table, dimension schema
    - normalization
    - Snowflake — only this one
    - Data cube

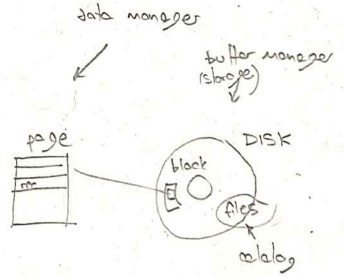
3NF

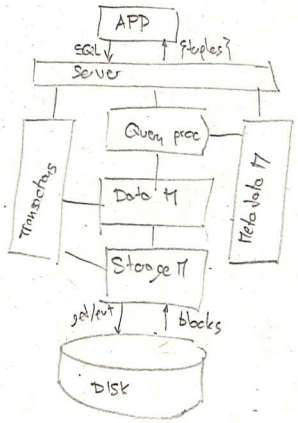
* every non-prime attribute is non-transitively dependent on every key of R

9 DATABASE SYSTEM

- Reasons — redundancy, inconsistency
- Tasks — extracting SQL tuples
- Memory hierarchy
- Storage manager
  - control access to external storage
  - management of files and blocks (catalogue)
  - buffer management
- Storing data
  - structure of record
    - fixed / variable length — pro (length, ptr)
    - header — RID (record identifier)
    - slotted page structure
    - TID — ↳ row chaining
- Data manager
  - BLOB (large obj)
  - IOT (index org. tables)
  - table space
- Buffer management → replacement policies
- Meta-data management — OS → double page fault

B+, R trees





10 QUERY PROCESSING ALGORITHMS

- machine code
- relational algebra
- Basics
  - query selectivity — $\frac{\#\text{tuples out}}{\#\text{tuples input table}}$
  - attribute cardinality → how many values
  - indexes
- Table access
  - scan
  - index scan = seek

cluster = leaves contain data, not ptr

- Sorting
  - 2 phase external sorting — $O(n \log n)$
    - memory fitting
    - M N — N # page input, M memory buffer size
    - 1. create runs
    - 2. merge runs
    - special cases
  - multi-pass

$M \geq N$ (no merge → 1 step)
$M < \sqrt{N}$ (multiple phases)

$x) = m\left(1 - \left(1 - \frac{1}{m}\right)^k\right)$

↳ # of blocks accessed out of M total

- Joins NLJ → BNLJ
  - nested loop join (2 nested loops) $O(|R||S|) = O(N^2)$
  - canonical (classic) hash join $O(|R|+|S|) = O(N)$
  - grace hash join GHJ
  - partitioned (hybrid) hash join
  - multi pass radix partitioning
  - parallel radix join
- Group by
  - sorting (+ merging)
  - hashing (+ partitioning)
- Sort vs Hashing

block/hash =

BNLJ

$i = \log_T P$

SMJ

• sort merge join

partitioning (before)
merging (after)

11 QUERY OPTIMIZATION

- Plan
- execution models
  - iterator implements operation
  - dataflow operations
  - pro/cons
- Selection operation
  - file scan
  - A1 (linear search) b_r
  - A2 (binary search) on ordered $b_r/2$ (key) $\log_2 b_r$ + all satisfy
- Selection using indexes
  - A3 (primary index on candidate key, equality) HT; +1
  - A4 (primary index on non-key, equality) HT; + # blocks containing result
  - A5 (equality on search-key of secondary index)
- Selection with comparison $\sigma_{A \geq V}(r)$
  - A6 (primary index, comparison)
  - A7 (secondary index, comparison)
- Complex Selection
  - A8 (conjunctive selection using one index) (use A1-7)
  - A9 (conjunctive selection using multiple-key index)
  - A10 (conjunctive selection by intersection of identifiers)
  - A11 (disjunctive selection by union of identifiers)
  - Negation $\sigma_{\neg\theta}(r)$

[conj: $\sigma_{\theta_1} \wedge \dots \wedge \sigma_{\theta_n}$
disj: $\vee$ $\vee$

- Generating equivalent plans
  - enumeration
  - algorithms (generally undecidable)
    - dynamic programming
    - greedy heuristics
    - randomized algorithms
- Optimizations
  - early
- Plans for joins

• union (no dupls)
• select
• project
• join (NLJ / index)

b_r = # blocks for relation r

primary/secondary index ?
(stored directly on leaf?)

Transaction Manager

RULES

$\sigma_{\theta_1 \wedge \theta_2}(E) \equiv \sigma_{\theta_1}(\sigma_{\theta_2}(E))$

$(E_1 \bowtie_{\theta_1} E_2) \bowtie_{\theta_2 \wedge \theta_3} E_3 = E_1 \bowtie_{\theta_1 \wedge \theta_3} (E_2 \bowtie_{\theta_2} E_3)$

$\Pi_{L_1 \cup L_2}(E_1 \bowtie E_2) = \Pi_{L_1 \cup L_2}(\Pi_{L_1 \cup L_3} E_1 \bowtie \Pi_{L_2 \cup L_4} E_2)$

$\sigma_\theta(E_1 - E_2) = \sigma(E_1) - E_2$

?

- R join S
- three way join
- Order by / group by
- Heuristic Optimization
  - selection/projection early
  - most restrictive first

12-14

12 TRANSACTION PROCESSING

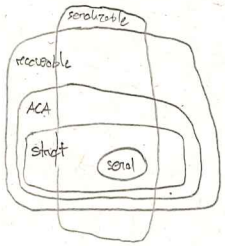
- Transaction
  - definition ordered sequence of read/write, commits/aborts
- Concurrency control respect in-order transaction, but not between them
- Operations
  - BOT (begin of transaction)
  - Commit
  - Abort
  - Read/write

$a <_T b$: a happens before b in po. T

r/w (total order) C/A ×

- ACID
  - atomicity
  - consistency
  - isolation
  - durability (recoverable)
- Abort / Commit
- CCT (concurrency control theory)
  - History H $po, <_H$ → respects order of transactions → total order
    - conflicting operation (both act on same item, one is write)
    - serial (-izable)
    - equivalence = same trans/op conflicts or read same way
  - Recovery theory
    - abort of single transaction R1 (undo)
    - system crash
      - main memory < R2 (redo committed), R3 (undo active)
      - disk
    - R1-4 R_4 : read backup
    - Undo/Redo (by transactions)
    - History type
      - recoverable (RC) $c_2 <_H c_1$
      - avoids cascading aborts (ACA) → otherwise thrashing
      - Strict (ST) $c_2 <_H op_1(A)$ $c_2 <_H r_1(A)$
- Database locking
  - 2PL (2 phase locking protocol) lock to access obj → GROWTH → SHRINK
    - strict (keep lock until EOT) - cascading
- Snapshot isolation (SI)
  - concurrency and availability
    - overhead, skew
  - definition of history
    - 2PL, SI

ASG (acyclic serializable graph)



deadlocks

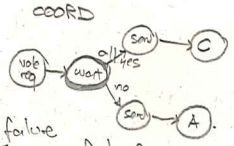
13 ATOMIC COMMIT (ACP)

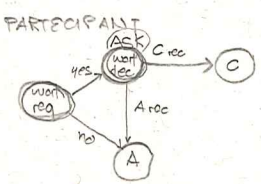
- consensus — on non-reliable system / no diff. between • slow system • failure
- properties
  - AC1-5
- 2PC (2 phase commit protocol)
  - protocol — commit req, commit phase
  - correctness
  - Coordinator
  - Participant — blocking
  - timeout — ACK → uncertain period
  - log record
- 3PC
  - NB (non-blocking) rule — • site failure • " & comm. failure
- Termination protocol
  - TR1-4
  - communication failures (not supported)
  - state req
  - quorum

1 → same decision
2 → no changing
3 → commit only if all
4 → no fail implies commit
5 → each decision, liveliness ← | 2PC problem

FAIL
• site
• comm

2n messages — 3n
2n rounds — 3





TR1 — some abort, then abort
TR2 — some comm, then comm
TR3 — all uncertain → abort
TR4 — some committable but no commit, then send pre-commit

C
A
uncertain committable

14 REPLICATION

- RAID
- advantages
  - performance — location transparency, distributed system
  - fault tolerance
  - application type — OLTP, OLAP
- Synchronous
- Asynchronous
  - pro/con
- Primary copy
- Update everywhere
  - pro/con
- Matrix result
  - feasibility
- Replication protocols
  - cost
  - write all-available copies
  - read/write/validation
- quorum protocol — set of sites, non-empty intersection
  - consensus protocol
  - read/write quorum
  - deals with partitioned
- main problems
- reconciliation
- version number

	perm	upd. everyw
sync	+ / - slowest	+ elegant / - not scalable
async	+ fast / - incons. → transfer	+ decentralized / - incons. not feasable

S: data replicated
w: fraction of updates
(ws: replication factor)
n: # nodes

each copy has weight w_i
$N = \sum w_i$
RT, WT } read/write thresholds
$2WT > N$
$RT + WT > N$

computing power $= \frac{n}{1 + w \cdot S (n-1)}$?

- Schema
- Deployment
- Replication strategy
  - The good
  - the bad
  - the ugly

16 DATABASE SECURITY

- Confidentiality
- Integrity
- SQL injection

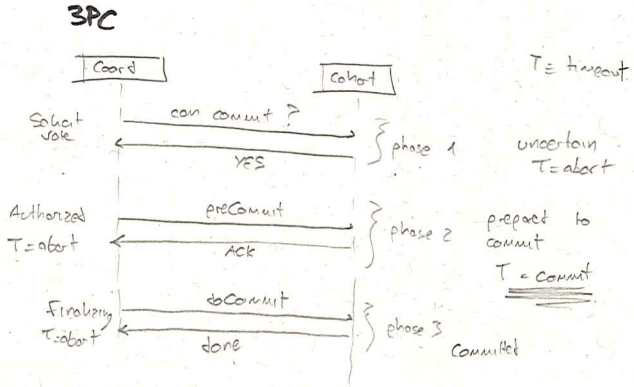
• authentication
• authorization
• auditing

privacy
leaking info: $P(S|Q) = P(S)$

GRANT permission

17 DB NEW HARDWARE

- (omitted)



15 KEY-VALUE STORE (KVS)

SQL cheatsheet

```
--DDL
CREATE TABLE Professor(
    id      INTEGER NOT NULL,
    name    VARCHAR(20),
    level   CHARACTER(2) DEFAULT "AP"
    lecture VARCHAR FOREIGN KEY REFERENCES TABLE Lectures       // same attribute name
                    FOREIGN KEY REFERENCES TableName(attribute)
    age NUMERIC (3,0) -- 3 digits, 0 of which after ","
        INT CHECK (age >=0 AND age<150)
    CONSTRAINT constraint_name CHECK person1<person2            PRIMARY KEY (ATT1, ATTN)
);

ALTER TABLE Professor ADD COLUMN(age INTEGER)
DROP TABLE Professor;

CREATE INDEX myindex ON Professor(name, age);
DROP INDEX myindex;
```

CREATE / ALTER / DROP — TABLE, INDEX, VIEW

```
--DML
INSERT INTO Student(legi, name) VALUES (29121, 'Frey'), ...;
INSERT INTO attends (Legi, 'Databases')
    SELECT legi                 } query
    FROM Student
    WHERE semester > 2

DELETE Student WHERE semester>13
UPDATE Student SET semester = semester+1

--views
CREATE VIEW testview AS (
    SELECT legi ...;
DROP VIEW testview;
```

INSERT into · UPDATE set · DELETE where

```
--QL
SELECT id, name
SELECT DISTINCT level
SELECT persid AS personid

SELECT ... UNION / INTERSECT / MINUS ...
SELECT AVG(), MAX(), MIN(), COUNT(), SUM()

FROM Professor,
     Student s
     JOIN Lecture L ON ...

WHERE level = 'FP'
    AND persid = id
WHERE NOT EXISTS (SELECT...) -- boolean
WHERE id NOT IN (SELECT...)
WHERE semester > ALL(SELECT semester FROM Student...)
WHERE sal > ALL (200, 300, 400) -- rewrite in -- WHERE sal > 200 AND sal > 300 AND sal > 400

WHERE semester >=1 AND semester <=6
WHERE semester BETWEEN 1 AND 6
WHERE semester IN (2,4,6)
```

SELECT / FROM / WHERE / ORDER/GROUP — JOIN


```
ORDER BY level DESC, name ASC

GROUP BY Professor.id
HAVING AVG(CP)>=3
--universal quantification
HAVING COUNT(*) = (SELECT COUNT(*) FROM Lecture) -- attends all lectures

WITH tablename AS (SELECT...) --view
-- query

CASE WHEN grade >=5 THEN 'sehr gut'
     WHEN ....  THEN
     ELSE ....
END)

WHERE name LIKE 'Prefix%'
           LIKE '_1'

--CONNECT BY
SELECT prerequisite FROM requires
CONNECT BY follow up = PRIOR prerequisite
START WITH follow-up = (SELECT...)
```

DESC: A Z … ASC: A Z …

HAVING / WITH AS / CASE / CONNECT BY / START WITH (exotic)

SIMONE GUGGIARI
ETH ZÜRICH 2016
LEGI # 14-932-867